马士兵教育Python 全栈文档

第二章：咚宝商城项目-项目开发

1、用户注册(后端)

1、发送短信

短信服务(Short Message Service)提供专业短信送达服务,支持快速发送短信验证码、短信通知等。 知名的有：腾讯短信服务，阿里云短信服务等等!

注意：

1、发短信不能重复发

2、短信验证码内容需要保存起来：redis

2、Flask-limiter限流

Flask-Limiter为flask路由提供了速率限制的特性。它支持使用内存、redis或者memcache作为存储环境的后端实现。

```
'''
默认的限制器
key_func参数是判断函数,表示以何种条件判断算一次访问?这里使用的是get_remote_address,此函数返回的是客户端的访问地址.
default_limits 是一个数组,用于依次提同判断条件.比如100/day是指一天100次访问限制.
常用的访问限制字符串格式如下:
10 per hour
10/hour
10/hour;100/day;2000 per year
100/day, 500/7days
注意默认的限制器对所有视图都有效,除非你自定义一个限制器用来覆盖默认限制器,或者使用limiter.exempt装饰器来取消限制
"""

limiter = Limiter(
    app,
    key_func=get_remote_address,
    default_limits=["1 per day", "2 per hour"]
)
```

基于FBV的简单使用

```
@app.route("/slow")
@limiter.limit("1 per day")
def slow():
    return "24"
```

基于CBV的限频方式

```
app = Flask(__name__)
limiter = Limiter(app, key_func=get_remote_address)

class MyView(flask.views.MethodView):
    decorators = [limiter.limit("10/second")]  # 重点，属性名字不能修改
    def get(self):
        return "get"

    def put(self):
        return "put"
```

配置

```
# 采用Redis保存数据，默认是内存，需要安装flask-redis
RATELIMIT_STORAGE_URL = 'redis://127.0.0.1:6379/0'
# 限制策略：移动窗口：时间窗口会自动变化
RATELIMIT_STRATEGY = 'moving-window'
```

3、Redis数据库介绍

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

一、NoSQL：一类新出现的数据库(not only sql)

- 泛指非关系型的数据库
- 不支持SQL语法
- 存储结构跟传统关系型数据库中的那种关系表完全不同，nosql中存储的数据都是KV形式
- NoSQL的世界中没有一种通用的语言，每种nosql数据库都有自己的api和语法，以及擅长的业务场景
- NoSQL中的产品种类相当多：
  - Redis
  - Mongodb
  - Hbase hadoop
  - Cassandra hadoop

NoSQL和SQL数据库的比较：

- 适用场景不同：sql数据库适合用于关系特别复杂的数据查询场景，nosql反之
- **事务** 特性的支持：sql对事务的支持非常完善，而nosql基本不支持事务
- 两者在不断地取长补短，呈现融合趋势

Redis特性

- Redis 与其他 key - value 缓存产品有以下三个特点：
- Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。
- Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
- Redis支持数据的备份，即master-slave模式的数据备份。

Redis应用场景

- 用来做缓存(ehcache/memcached)——redis的所有数据是放在内存中的（内存数据库）
- 可以在某些特定应用场景下替代传统数据库——比如社交类的应用
- 在一些大型系统中，巧妙地实现一些特定的功能：session共享、购物车
- 只要你有丰富的想象力，redis可以用在可以给你无限的惊喜.....

Redis 安装

- 我们采用的redis稳定版本是4.0.9
- step1:下载

  > wget http://download.redis.io/releases/redis-4.0.9.tar.gz

- step2:解压

> tar xzf redis-4.0.9.tar.gz

- step3:移动，放到usr/local目录下

> mv ./redis-4.0.9 /usr/local/redis/

- step4:进入redis目录

> cd /usr/local/redis/

- step5:生成

> make

```
[root@mylinux1 redis-4.0.11]# make
cd src && make all
make[1]: 进入目录“ /root/redis-4.0.11/src”
    CC adlist.o
In file included from adlist.c:34:0:
zmalloc.h:50:31: 致命错误：jemalloc/jemalloc.h：没有那个文件或目录
 #include <jemalloc/jemalloc.h>
                               ^
编译中断。
make[1]: *** [adlist.o] 错误 1
make[1]: 离开目录“ /root/redis-4.0.11/src”
make: *** [all] 错误 2
```

> 可以执行命令：make MALLOC=libc

- step6:测试,这段运行时间会较长

> make test

```
[root@mylinux1 redis-4.0.11]# make test
cd src && make test
make[1]: 进入目录“ /root/redis-4.0.11/src”
You need tcl 8.5 or newer in order to run the Redis test
make[1]: *** [test] 错误 1
make[1]: 离开目录“ /root/redis-4.0.11/src”
make: *** [test] 错误 2
[root@mylinux1 redis-4.0.11]#
```

> 先运行命令: yum install tcl

- step7:安装,将redis的命令安装到`/usr/local/bin/`目录

> make install

- step8:安装完成后，我们进入目录`/usr/local/bin`中查看

> cd /usr/local/bin ls -all

  - redis-server redis服务器
  - redis-cli redis命令行客户端
  - redis-benchmark redis性能测试工具
  - redis-check-aof AOF文件修复工具
  - redis-check-rdb RDB文件检索工具

- step9:配置文件，移动到`/etc/`目录下
  - 配置文件目录为`/usr/local/redis/redis.conf`

> cp /usr/local/redis/redis.conf /etc/redis/

Redis配置

- 绑定ip：如果需要远程访问，可将此行注释，或绑定一个真实ip

  > bind 127.0.0.1

- 端口，默认为6379

  > port 6379

- 是否以守护进程运行
  - 如果以守护进程运行，则不会在命令行阻塞，类似于服务
  - 如果以非守护进程运行，则当前终端被阻塞
  - 设置为yes表示守护进程，设置为no表示非守护进程
  - 推荐设置为yes

  > daemonize yes

- 数据文件

  > dbfilename dump.rdb

- 数据文件存储路径

  > dir /var/lib/redis

- 日志文件

  > logfile "/var/log/redis/redis-server.log"

- 数据库，默认有16个

  > database 16

服务器命令

服务器端的命令为redis-server

redis-server --help

ps aux | grep redis 查看redis服务器进程 kill -9 pid 杀死redis服务器 redis-server /etc/redis/redis.conf 指定加载的配置文件

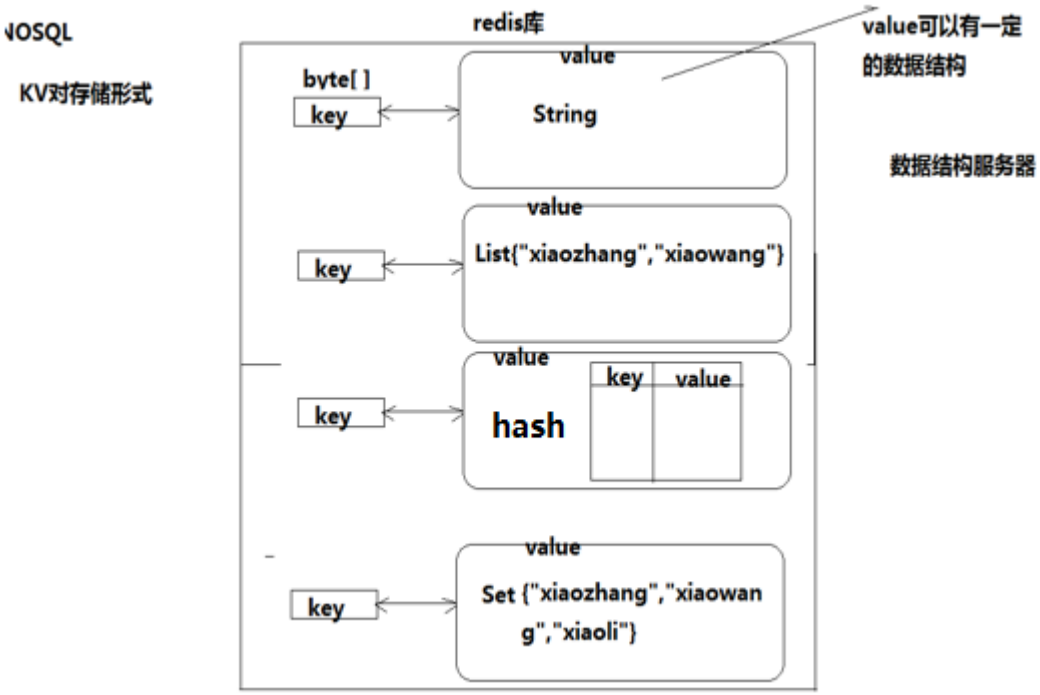
数据操作命令

- 客户端的命令为redis-cli
- 数据库没有名称，默认有16个，通过0-15来标识，连接redis默认选择第一个数据库

  > select 10

数据结构

- redis是key-value的数据结构，每条数据都是一个键值对
- 键的类型是字符串
- 注意：键不能重复



- 值的类型分为五种：
  - 字符串string
  - 哈希hash
  - 列表list
  - 集合set
  - 有序集合zset

4、Redis各种数据类型操作命令

1、string类型

- 字符串类型是 Redis 中最为基础的数据存储类型，它在 Redis 中是二进制安全的，这便意味着该类型可以接受任何格式的数据，如JPEG图像数据或Json对象描述信息等。在Redis中字符串类型的Value最多可以容纳的数据长度是512M。

保存

如果设置的键不存在则为添加，如果设置的键已经存在则修改

- 设置键值

  > set key value

- 例1：设置键为 `name` 值为 `laoxiao` 的数据

  > set name laoxiao

  设置键值及过期时间，以秒为单位

  > setex key seconds value

- 例2：设置键为 `aa` 值为 `aa` 过期时间为3秒的数据

  > setex aa 3 aa

```
127.0.0.1:6379> setex aa 3 aa
OK
127.0.0.1:6379> get aa
"aa"
127.0.0.1:6379> get aa
(nil)
```

- 设置多个键值

> mset key1 value1 key2 value2 ...

- 例3：设置键为 `a1` 值为 `python` 、键为 `a2` 值为 `java` 、键为 `a3` 值为 `c`

> mset a1 python a2 java a3 c

```
127.0.0.1:6379> mset a1 python a2 java a3 c
OK
127.0.0.1:6379> get a1
"python"
127.0.0.1:6379> get a2
"java"
127.0.0.1:6379> get a3
"c"
127.0.0.1:6379>
```

- 追加值

> append key value

- 例4：向键为 `a1` 中追加值 `haha`

> append a1 haha

```
127.0.0.1:6379> append a1 haha
(integer) 10
127.0.0.1:6379> get a1
"pythonhaha"
```

获取

- 获取：根据键获取值，如果不存在此键则返回 `nil`

> get key

- 例5：获取键 `name` 的值

> get name

- 根据多个键获取多个值

> mget key1 key2 ...

- 例6：获取键 `a1、a2、a3` 的值

> mget a1 a2 a3

```
127.0.0.1:6379> mget a1 a2 a3
1) "python"
2) "java"
3) "c"
127.0.0.1:6379>
```

2、键命令

- 查找键，参数支持正则表达式

  > keys pattern

- 例1：查看所有键

  > keys *

```
127.0.0.1:6379> keys *
1) "a2"
2) "c++"
3) "a1"
4) "java"
5) "a3"
6) "name"
7) "code"
```

- 例2：查看名称中包含`a`的键

  > keys a*

```
127.0.0.1:6379> keys a*
1) "a2"
2) "a1"
3) "a3"
```

- 判断键是否存在，如果存在返回`1`，不存在返回`0`

  > exists key1

- 例3：判断键`a1`是否存在

  > exists a1

```
127.0.0.1:6379> EXISTS a1
(integer) 1
127.0.0.1:6379> EXISTS a2
(integer) 1
127.0.0.1:6379> EXISTS cc
(integer) 0
```

- 查看键对应的`value`的类型

  > type key

- 例4：查看键`a1`的值类型，为redis支持的五种类型中的一种

  > type a1

```
127.0.0.1:6379> type a1
string
```

- 删除键及对应的值

  > del key1 key2 ...

- 例5：删除键`a2、a3`

  > del a2 a3

```
127.0.0.1:6379> mget a1 a2 a3
1) "pythonhaha"
2) "java"
3) "c"
127.0.0.1:6379> del a2 a3
(integer) 2
127.0.0.1:6379> mget a1 a2 a3
1) "pythonhaha"
2) (nil)
3) (nil)
```

- 设置过期时间，以秒为单位
- 如果没有指定过期时间则一直存在，直到使用`DEL`移除

> expire key seconds

- 例6：设置键`a1`的过期时间为3秒

> expire a1 3

```
127.0.0.1:6379> EXPIRE a1 3
(integer) 1
127.0.0.1:6379> get a1
"pythonhaha"
127.0.0.1:6379> get a1
(nil)
```

- 查看有效时间，以秒为单位

> ttl key

- 例7：查看键`bb`的有效时间

> ttl bb

```
127.0.0.1:6379> setex bb 10 bb
OK
127.0.0.1:6379> ttl bb
(integer) 7
```

3、hash类型

- **hash**用于存储对象，对象的结构为属性、值
- **值**的类型为**string**

增加、修改

- 设置单个属性

> hset key field value

- 例1：设置键`user`的属性`name`为`itheima`

> hset user name itheima

- 设置多个属性

> hmset key field1 value1 field2 value2 ...

- 例2：设置键`u2`的属性`name`为`laoxiao`、属性`age`为`11`

> hmset u2 name laoxiao age 11

- 获取指定键所有的属性

> hkeys key

- 例3：获取键u2的所有属性

> hkeys u2

!

- 获取一个属性的值

> hget key field

- 例4：获取键`u2`属性`name`的值

> hget u2 name

- 获取多个属性的值

> hmget key field1 field2 ...

- 例5：获取键`u2`属性`name`、`age`的值

> hmget u2 name age

- 获取所有属性的值

> hvals key

- 例6：获取键`u2`所有属性的值

> hvals u2

删除

- **删除整个hash键及值，使用del命令**
- 删除属性，属性对应的值会被一起删除

> hdel key field1 field2 ...

- 例7：删除键`u2`的属性`age`

> hdel u2 age

可能出现的错误

```
127.0.0.1:6379> hset user name itheima
(error) MISCONF Redis is configured to save RDB snapshots, but is currently not
able to persist on disk. Commands that may modify the data set are disabled. Ple
ase check Redis logs for details about the error.
127.0.0.1:6379> config set stop-writes-on-bgsave-error no
OK
127.0.0.1:6379> hset user name itheima
(integer) 1
```

> MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk. Commands that may modify the data set are disabled. Please check Redis logs for details about the error.

> Redis被配置为保存数据库快照，但它目前不能持久化到硬盘。用来修改集合数据的命令不能用

- 原因：
  - 强制关闭Redis快照导致不能持久化。
- 解决方案：
  - 运行config set stop-writes-on-bgsave-error no　命令后，关闭配置项stop-writes-on-bgsave-error解决该问题。

4、list类型

- 列表的元素类型为string
- 按照插入顺序排序

增加

- 在左侧插入数据

  > lpush key value1 value2 ...

- 例1：从键为`a1`的列表左侧加入数据`a 、 b 、c`

  > lpush a1 a b c

- 在右侧插入数据

  > rpush key value1 value2 ...

- 例2：从键为`a1`的列表右侧加入数据`0、1`

  > rpush a1 0 1

- 在指定元素的前或后插入新元素

  > linsert key before或after 现有元素 新元素

- 例3：在键为`a1`的列表中元素`b`前加入`3`

  > linsert a1 before b 3

- 返回列表里指定范围内的元素
  - `start`、`stop`为元素的下标索引
  - 索引从左侧开始，第一个元素为0
  - 索引可以是负数，表示从尾部开始计数，如`-1`表示最后一个元素

  > lrange key start stop

- 例4：获取键为`a1`的列表所有元素

  > lrange a1 0 -1

设置指定索引位置的元素值

- 索引从左侧开始，第一个元素为0
- 索引可以是负数，表示尾部开始计数，如`-1`表示最后一个元素

  > lset key index value

- 例5：修改键为`a1`的列表中下标为`1`的元素值为`z`

> lset a 1 z

删除

- 删除指定元素
  - 将列表中前 `count` 次出现的值为 `value` 的元素移除
  - count > 0: 从头往尾移除
  - count < 0: 从尾往头移除
  - count = 0: 移除所有

> lrem key count value

- 例6.1：向列表 `a2` 中加入元素 `a、b、a、b、a、b`

> lpush a2 a b a b a b

- 例6.2：从 `a2` 列表右侧开始删除2个 `b`

> lrem a2 -2 b

- 例6.3：查看列表 `a2` 的所有元素

> **lrange a2 0 -1**

5、set类型

- 无序集合
- 元素为string类型
- 元素具有唯一性，不重复
- 说明：对于集合没有修改操作

增加

- 添加元素

> sadd key member1 member2 ...

- 例1：向键 `a3` 的集合中添加元素 `zhangsan`、`lisi`、`wangwu`

> sadd a3 zhangsan sili wangwu

获取

- 返回所有的元素

> smembers key

- 例2：获取键 `a3` 的集合中所有元素

> smembers a3

删除

- 删除指定元素

> srem key

- 例3：删除键 `a3` 的集合中元素 `wangwu`

> srem a3 wangwu

6、zset类型

- sorted set，有序集合
- 元素为string类型
- 元素具有唯一性，不重复
- 每个元素都会关联一个double类型的score，表示权重，通过权重将元素从小到大排序
- 说明：没有修改操作

增加

- 添加

  > zadd key score1 member1 score2 member2 ...

- 例1：向键`a4`的集合中添加元素`lisi`、`wangwu`、`zhaoliu`、`zhangsan`，权重分别为`4、5、6、3`

  > zadd a4 4 lisi 5 wangwu 6 zhaoliu 3 zhangsan

获取

- 返回指定范围内的元素
- start、stop为元素的下标索引
- 索引从左侧开始，第一个元素为0
- 索引可以是负数，表示从尾部开始计数，如`-1`表示最后一个元素

  > zrange key start stop

- 例2：获取键`a4`的集合中所有元素

  > zrange a4 0 -1

- 返回`score`值在`min`和`max`之间的成员

  > zrangebyscore key min max

- 例3：获取键`a4`的集合中权限值在`5和6之间`的成员

  > zrangebyscore a4 5 6

- 返回成员`member`的`score`值

  > zscore key member

- 例4：获取键`a4`的集合中元素`zhangsan`的权重

  > zscore a4 zhangsan

删除

- 删除指定元素

  > zrem key member1 member2 ...

- 例5：删除集合`a4`中元素`zhangsan`

  > zrem a4 zhangsan

- 删除权重在指定范围的元素

> zremrangebyscore key min max

- 例6：删除集合`a4`中权限在`5、6之间`的元素

> zremrangebyscore a4 5 6

5、与Python交互

安装包

pip install redis

- 引入模块

> from redis import StrictRedis

- 这个模块中提供了`StrictRedis对象`，用于连接redis服务器，并按照不同类型提供 了不同方法，进行交互操作

StrictRedis对象方法

- 通过init创建对象，指定参数host、port与指定的服务器和端口连接，host默认为localhost，port默认为6379，db默认为0

```
sr = StrictRedis(host='localhost', port=6379, db=0)

简写
sr=StrictRedis()
```

- 根据不同的类型，拥有不同的实例方法可以调用，与前面学的redis命令对应，方法需要的参数与命令的参数一致

string

- set
- setex
- mset
- append
- get
- mget
- key

keys

- exists
- type
- delete
- expire
- getrange
- ttl

hash

- hset
- hmset

- hkeys
- hget
- hmget
- hvals
- hdel

list

- lpush
- rpush
- linsert
- lrange
- lset
- lrem

set

- sadd
- smembers
- srem

zset

- zadd
- zrange
- zrangebyscore
- zscore
- zrem
- zremrangebyscore

6、Flask-redis

在 Flask 中使用 Redis 可以直接使用 `flask-redis` 支持包，它是对 `redis.py` 的扩展，使用起来非常方便。使用以下命令即可安装该支持包：

```
pip install flask-redis
flask-redis 的配置非常方便，只需要在配置文件中增加 REDIS_URL 的配置即可。
```

```
REDIS_URL = "redis://:password@localhost:6379/0"
flask-redis 初始化同样非常简单，只需要两行代码即可。

redis_client = FlaskRedis()
...
redis_cline.init_app(app)
```

7、提交注册信息

验证手机号 — 填写账号信息 注册成功

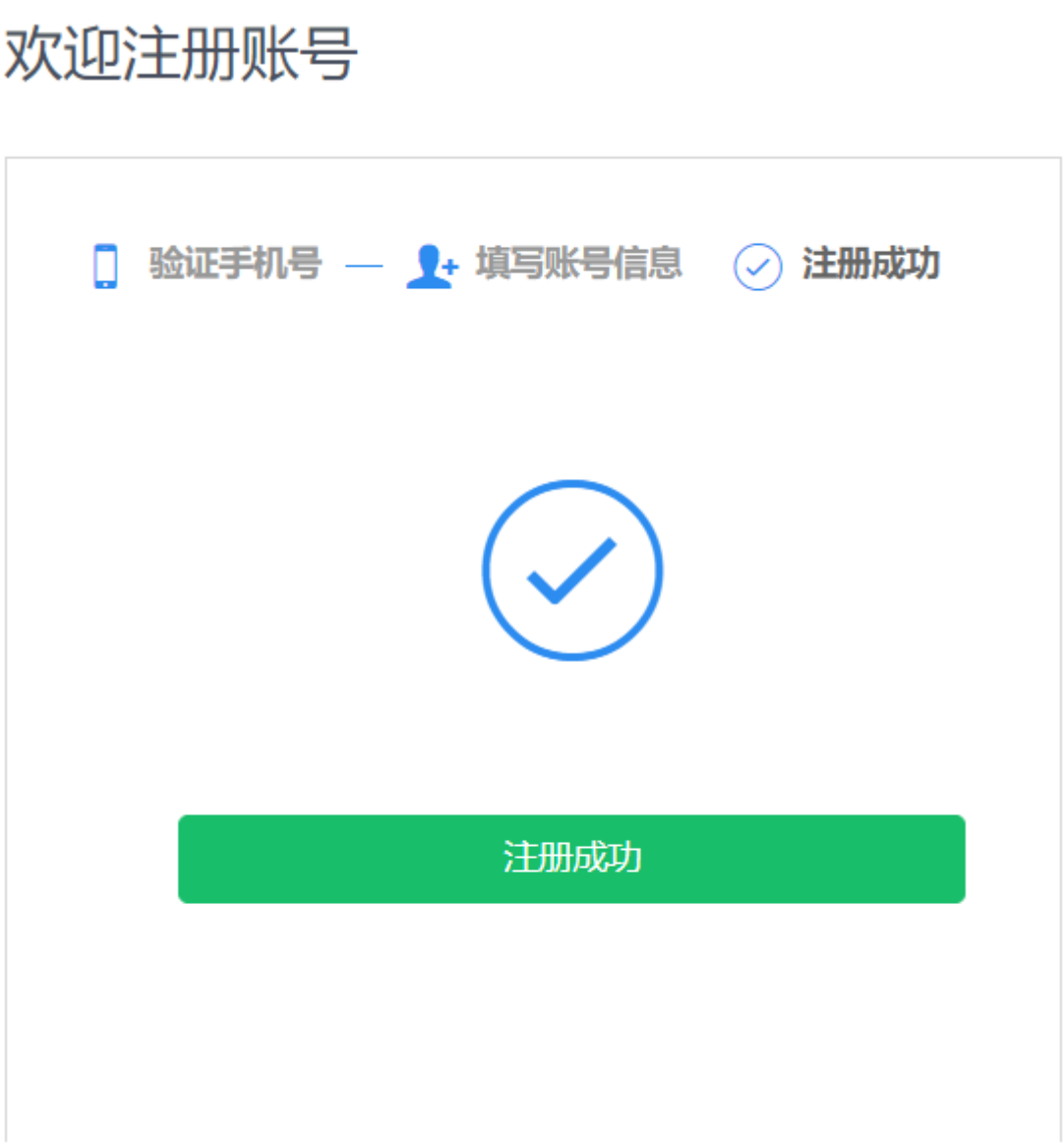
* 用户名 请输入你的姓名

* 邮箱 请输入你的邮箱

* 密码 请输入你的密码

确认密码 请再次输入你的密码

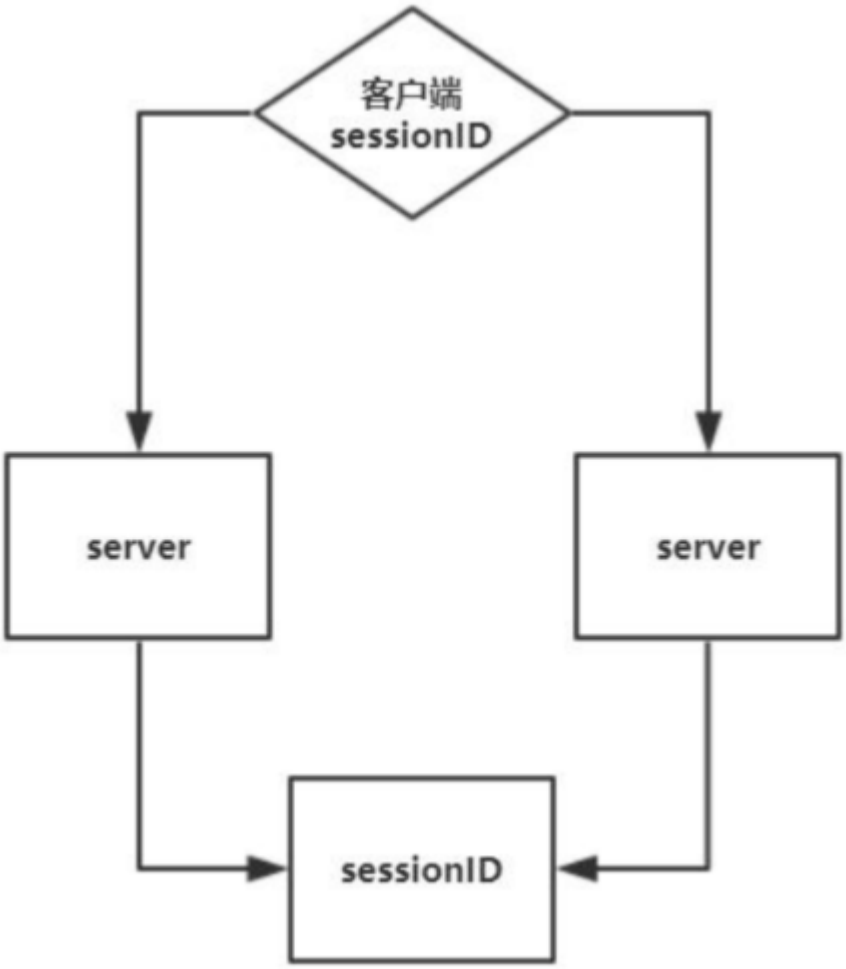
注册



2、Token的使用

1、登录

思考：在分布式集群的情况下，session数据的存储怎么解决?

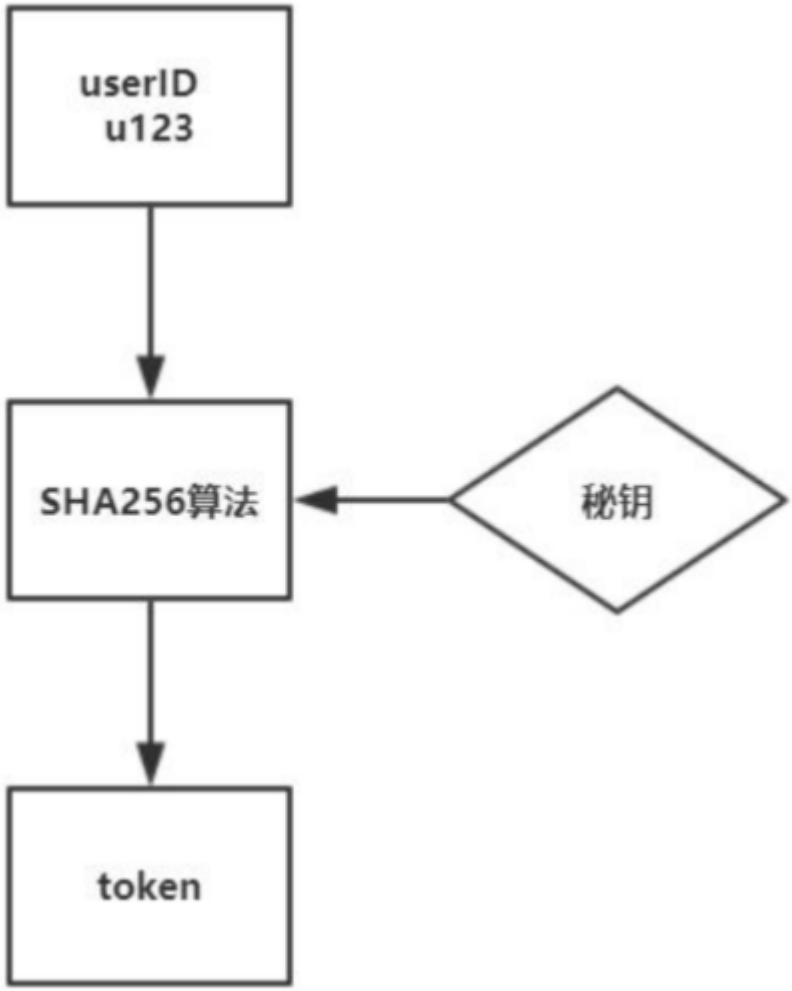


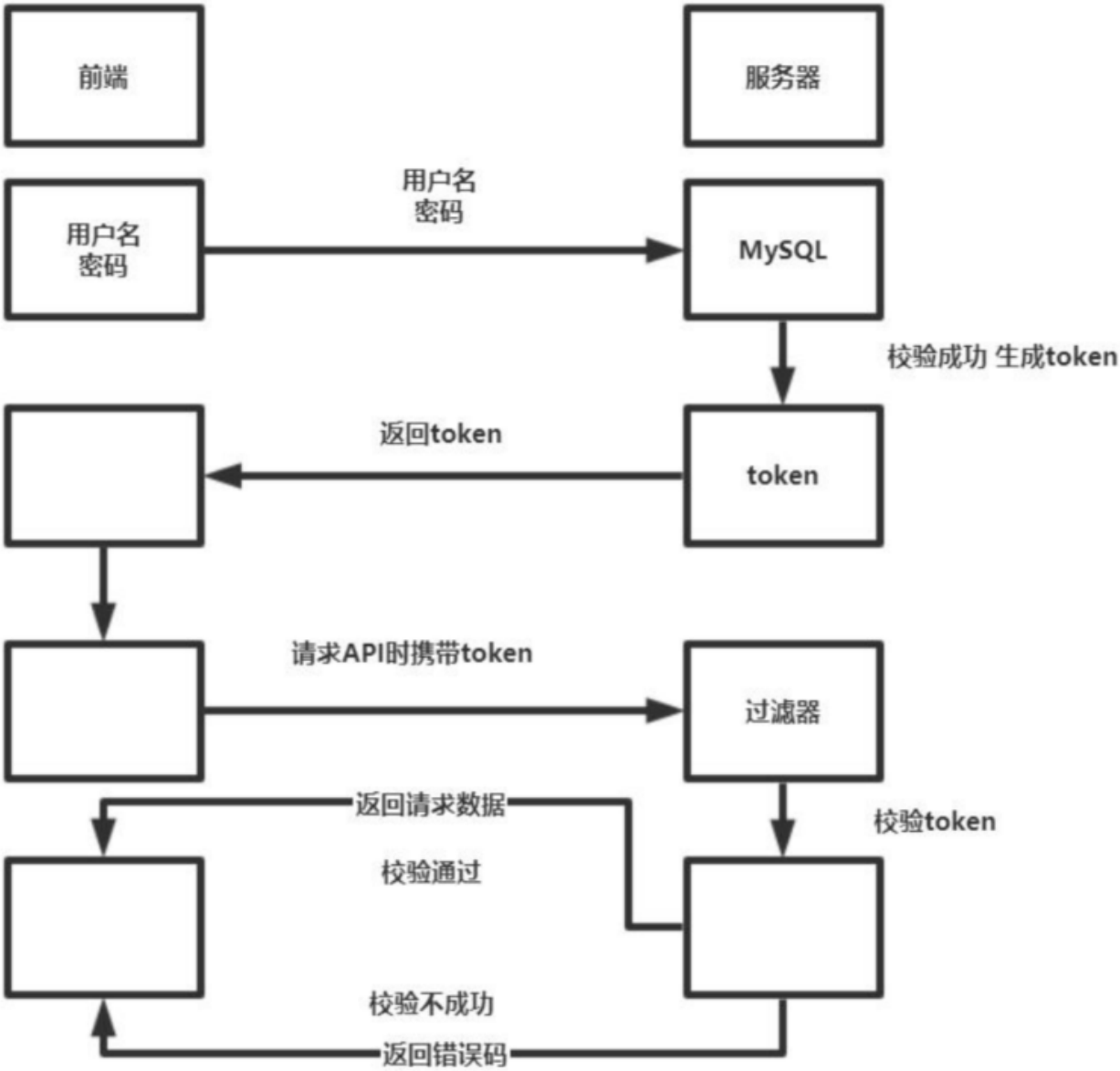
2、Token的原理

Json Web Token(JWT)

JSON Web Token（JWT）是一个非常轻巧的规范。这个规范允许我们使用JWT在两个组织之间传递安全可靠的信息。

基于 Token 的身份验证是无状态的，我们不将用户信息存在服务器中。这种概念解决了 在服务端存储信息时的许多问题。NoSession 意味着你的程序可以根据需要去增减机器，而 不用去担心Session数据的共享问题。


userID
u123
SHA256算法
秘钥
token



两种实现：

- itsdangerous模块中的TimedJSONWebSignatureSerializer
- pyjwt模块 : pip install pyjwt

3、验证Token的装饰器

当有业务访问时，必须登录过才可以访问时，为了避免每次编写代码，我们可以编写一个 装饰来验证用户是否登录。

1、定义请求钩子：

作用：可以在请求进来之前，先把token里面的